A - Bitcoin Price Prediction

2024-12-16

```
# Load necessary libraries
library(dplyr)

## Warning: package 'dplyr' was built under R version 4.4.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(lubridate)

## Warning: package 'lubridate' was built under R version 4.4.2

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.4.2

# Read the data
data <- read.csv("C:\\Users\\USER\\Downloads\\data1\\main.csv")

# Check the structure of the data
str(data)

## 'data.frame':    188317 obs. of  11 variables:
##  $ Open.Time                    : num  1.61e+12 1.61e+12 1.61e+12 1.61e+12
1.61e+12 ...
##  $ Open                         : num  28924 28962 29010 28990 28983 ...
##  $ High                         : num  28962 29018 29017 29000 28996 ...
##  $ Low                          : num  28913 28961 28974 28972 28972 ...
##  $ Close                        : num  28962 29010 28989 28983 28976 ...
##  $ Volume                       : num  27.5 58.5 42.5 30.4 24.1 ...
##  $ Close.Time                   : num  1.61e+12 1.61e+12 1.61e+12 1.61e+12
1.61e+12 ...
##  $ Quote.asset.volume           : num  794382 1695803 1231359 880017 699226
```

```
...
##  $ Number.of.trades            : int  1292 1651 986 959 726 952 750 782
886 1558 ...
##  $ Taker.buy.base.asset.volume : num  16.78 33.73 13.25 9.46 6.81 ...
##  $ Taker.buy.quote.asset.volume: num  485391 978176 384077 274083 197519
...
```

```
head(data)
```

```
##      Open.Time     Open     High      Low    Close   Volume   Close.Time
## 1 1.609459e+12 28923.63 28961.66 28913.12 28961.66 27.45703 1.609459e+12
## 2 1.609459e+12 28961.67 29017.50 28961.01 29009.91 58.47750 1.609459e+12
## 3 1.609459e+12 29009.54 29016.71 28973.58 28989.30 42.47033 1.609459e+12
## 4 1.609459e+12 28989.68 28999.85 28972.33 28982.69 30.36068 1.609459e+12
## 5 1.609459e+12 28982.67 28995.93 28971.80 28975.65 24.12434 1.609459e+12
## 6 1.609460e+12 28975.65 28979.53 28933.16 28937.11 22.39601 1.609460e+12
##   Quote.asset.volume Number.of.trades Taker.buy.base.asset.volume
## 1           794382.0             1292                   16.777195
## 2          1695802.9             1651                   33.733818
## 3          1231358.7              986                   13.247444
## 4           880016.8              959                    9.456028
## 5           699226.2              726                    6.814644
## 6           648322.7              952                    9.127550
##   Taker.buy.quote.asset.volume
## 1                     485390.8
## 2                     978176.5
## 3                     384076.9
## 4                     274083.1
## 5                     197519.4
## 6                     264217.9
```

Step 2: Convert Timestamps to Date-Time

```
# Convert Open Time and Close Time to POSIXct (date-time format)
data$Open.Time <- as.POSIXct(data$Open.Time / 1000, origin = "1970-01-01", tz
= "UTC")
data$Close.Time <- as.POSIXct(data$Close.Time / 1000, origin = "1970-01-01",
tz = "UTC")

# Verify the conversion
head(data$Open.Time)
```

```
## [1] "2021-01-01 00:00:00 UTC" "2021-01-01 00:01:00 UTC"
## [3] "2021-01-01 00:02:00 UTC" "2021-01-01 00:03:00 UTC"
## [5] "2021-01-01 00:04:00 UTC" "2021-01-01 00:05:00 UTC"
```

```
head(data$Close.Time)
```

```
## [1] "2021-01-01 00:00:59 UTC" "2021-01-01 00:01:59 UTC"
## [3] "2021-01-01 00:02:59 UTC" "2021-01-01 00:03:59 UTC"
## [5] "2021-01-01 00:04:59 UTC" "2021-01-01 00:05:59 UTC"
```

Step 3: Aggregate Data to Daily Intervals

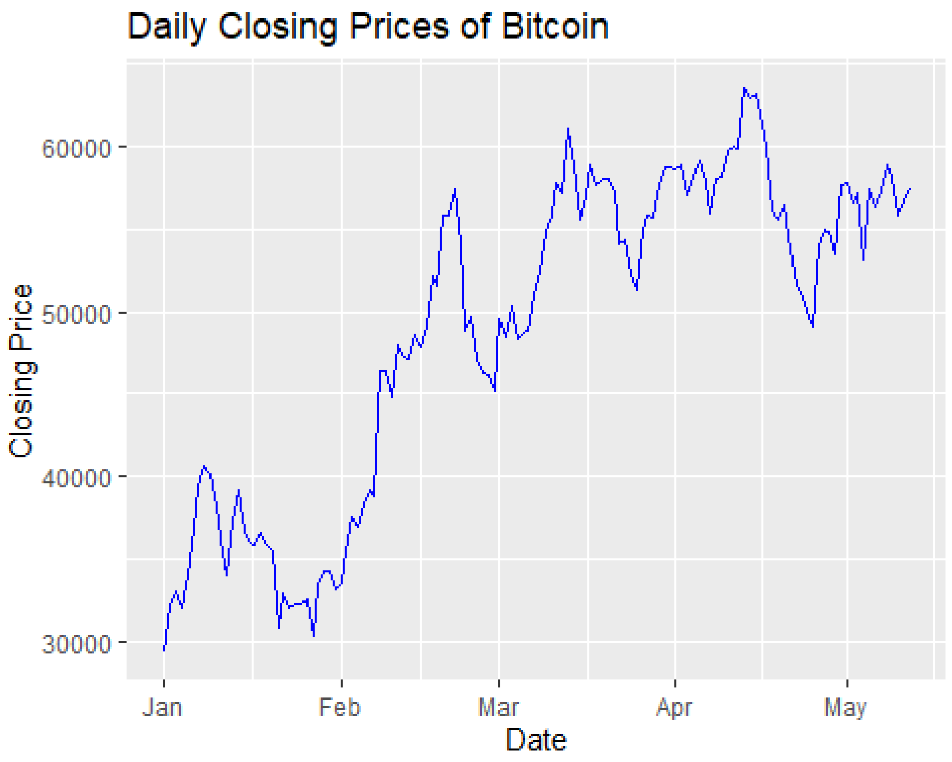
```
# Aggregate to daily data
daily_data <- data %>%
  mutate(Date = as.Date(Open.Time)) %>%
  group_by(Date) %>%
  summarize(
    Open = first(Open),
    High = max(High),
    Low = min(Low),
    Close = last(Close),
    Volume = sum(Volume),
    Trades = sum(Number.of.trades)
  )

# Verify the daily aggregated data
head(daily_data)
```

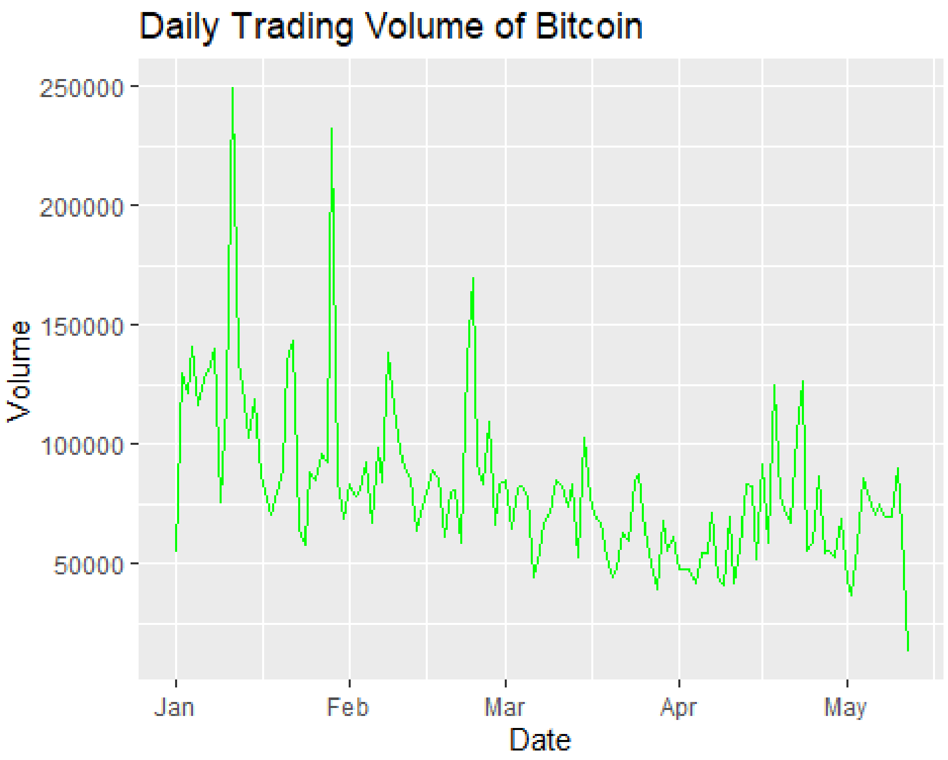
```
## # A tibble: 6 × 7
##   Date         Open   High    Low  Close  Volume  Trades
##   <date>      <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <int>
## 1 2021-01-01 28924. 29600  28625. 29332.  54183. 1314910
## 2 2021-01-02 29332. 33300  28947. 32178. 129994. 2245922
## 3 2021-01-03 32176. 34778. 31963. 33000. 120958. 2369698
## 4 2021-01-04 33000. 33600  28130  31989. 140900. 2642408
## 5 2021-01-05 31990. 34360  29900  33950. 116050. 2526851
## 6 2021-01-06 33950. 36939. 33288  36769. 127139. 2591783
```

Step 4: Visualize the Data

```
# Plot Daily Closing Prices
ggplot(daily_data, aes(x = Date, y = Close)) +
  geom_line(color = "blue") +
  labs(title = "Daily Closing Prices of Bitcoin", x = "Date", y = "Closing
Price")
```



```r
# Plot Daily Trading Volume
ggplot(daily_data, aes(x = Date, y = Volume)) +
  geom_line(color = "green") +
  labs(title = "Daily Trading Volume of Bitcoin", x = "Date", y = "Volume")
```



Step 5: Check for Stationarity

```
# Load required library
library(tseries)

## Warning: package 'tseries' was built under R version 4.4.2

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

# Perform the Augmented Dickey-Fuller (ADF) test on Closing Prices
cat("ADF Test for Closing Prices:\n")

## ADF Test for Closing Prices:

adf_result <- adf.test(daily_data$Close, alternative = "stationary")
print(adf_result)

##
##  Augmented Dickey-Fuller Test
##
## data:  daily_data$Close
## Dickey-Fuller = -2.1389, Lag order = 5, p-value = 0.5187
## alternative hypothesis: stationary

# Check stationarity and apply differencing if needed
if (adf_result$p.value > 0.05) {
  cat("\nSeries is non-stationary. Applying first differencing...\n")

  # Apply differencing and add to dataframe
  daily_data$Close_diff <- c(NA, diff(daily_data$Close))  # Prepend NA to
align rows

  # Verify the new column
  print(head(daily_data))

  # Perform ADF test on the differenced series
  cat("\nADF Test for Differenced Closing Prices:\n")
  adf_diff_result <- adf.test(na.omit(daily_data$Close_diff), alternative =
"stationary")
  print(adf_diff_result)

  # Check if stationarity is achieved
  if (adf_diff_result$p.value <= 0.05) {
    cat("\nThe differenced series is stationary.\n")
  } else {
    cat("\nThe differenced series is still non-stationary. Further
transformations may be required.\n")
  }
} else {
  cat("\nThe original series is stationary. No differencing is needed.\n")
}
```

```
##
## Series is non-stationary. Applying first differencing...
## # A tibble: 6 × 8
##   Date         Open   High    Low  Close  Volume  Trades Close_diff
##   <date>      <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <int>      <dbl>
## 1 2021-01-01 28924. 29600  28625. 29332.  54183. 1314910        NA
## 2 2021-01-02 29332. 33300  28947. 32178. 129994. 2245922      2847.
## 3 2021-01-03 32176. 34778. 31963. 33000. 120958. 2369698       822.
## 4 2021-01-04 33000. 33600  28130  31989. 140900. 2642408     -1011.
## 5 2021-01-05 31990. 34360  29900  33950. 116050. 2526851      1961.
## 6 2021-01-06 33950. 36939. 33288  36769. 127139. 2591783      2820.
##
## ADF Test for Differenced Closing Prices:
```

```
## Warning in adf.test(na.omit(daily_data$Close_diff), alternative =
## "stationary"): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(daily_data$Close_diff)
## Dickey-Fuller = -4.6466, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
##
```

```
##
## The differenced series is stationary.
```

Step 6: Fit ARIMA Model and Forecast

```r
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.4.2
```

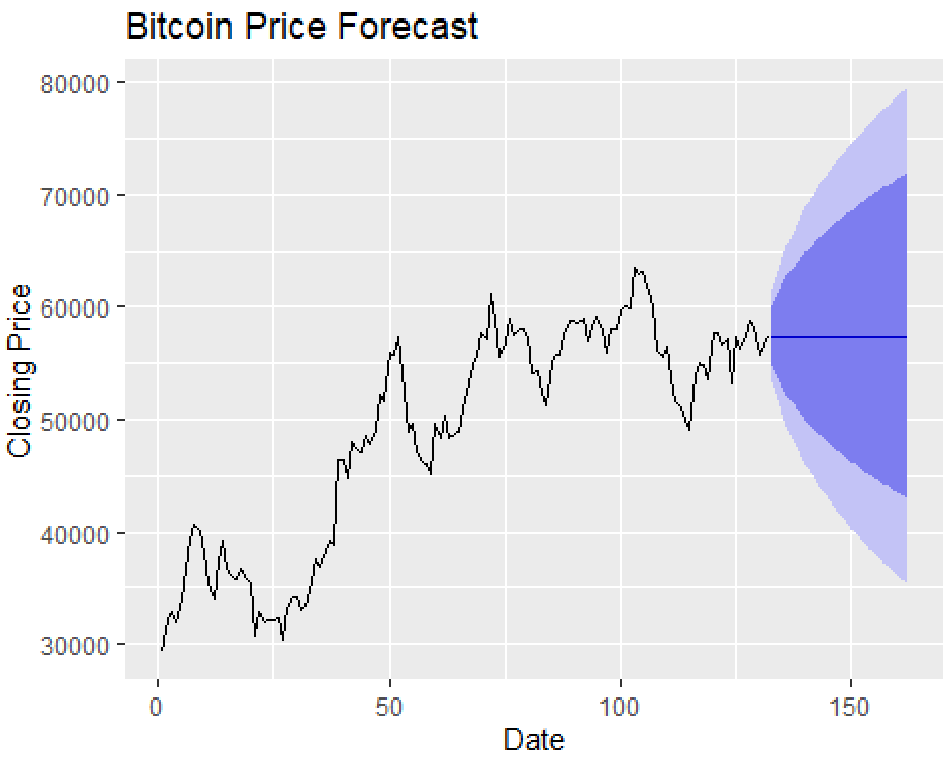
```r
# Fit ARIMA Model
fit <- auto.arima(daily_data$Close, seasonal = FALSE)

# Summary of the Model
summary(fit)
```

```
## Series: daily_data$Close
## ARIMA(0,1,0)
##
## sigma^2 = 4214274:  log likelihood = -1185.02
## AIC=2372.03   AICc=2372.07   BIC=2374.91
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 213.2411 2045.079 1542.794 0.4115671 3.270829 0.9925672
##                     ACF1
## Training set -0.05299381
```

```r
# Forecast the next 30 days
forecasted <- forecast(fit, h = 30)

# Plot the Forecast
autoplot(forecasted) +
  labs(title = "Bitcoin Price Forecast", x = "Date", y = "Closing Price")
```


Bitcoin Price Forecast
Closing Price
80000
70000
60000
50000
40000
30000
0
50
100
150
Date